V.11 Binary Search Algorithm →

Question:

	s			m				e	
	0	1	2	3	4	5	6	7	8	9
arr =	2,	4,	6,	9,	11,	12,	14,	20,	36,	48

target = 36.

Algorithm

Step ① - Find the middle element

Step ② - target > mid ⇒ Search right side.
else, search left side.

Step ③ - if, middle element == target.
then, ans.

The steps are repeated.

s m e

[12, 14, 20, 36, 48]

[20, 36, 48]

m

Therefore, the middle element == target element.

Time Complexity ↓

(*) Best case for Binary Search is when the first middle value is equal to the target element.

O(1)

(*) Worst case is O(log N)

Code:

```
public class BinarySearch {

    psvm(String[] args) {
    int[] arr = {-18, -12, -4, 0, 13, 22, 64, 89}
    int target = 64;
    int ans = binarySearch(arr, target);
    sout(ans);
    }

Static int binarySearch(int[] arr, int target)
    {
        int start = 0;
        int end = arr.length - 1;
        while (start <= end)
            {
            int mid = start + (end - start)/2;
            if (target < arr[mid])
                {
                end = mid - 1;
                }
            else if (target > arr[mid])
                {
                start = mid + 1;
                }
            else { return mid; }
        }
        return -1;
}
```

(*) Order Agnostic Binary Search :

```
// finding whether the array is Ascending / Descendi
// After declaration of start and end.
    boolean isAsc = arr[start] < arr[end];
while (start <= end)
        {
          int mid = start + (end - start) /2;
        if (arr[mid] == target)
            {
              return mid;
            }
        if (isAsc)
            {
                if (target < arr[mid])
                  {
                    end = mid -1;
                  }
                else
                    {
                      Start = mid + 1;
                    }
            }
          else {
                  if (target > arr[mid])
                  {
                    end = mid - 1;
                  }
                  else
                    {
                      start = mid + 1;
                    }
              }
      }
    return -1;
}
```